Red Hat Enterprise Linux Kernel Crypto API Cryptographic Module v4.0 with CPACF

FIPS 140-2 Non-Proprietary Security Policy

Version 1.1

Last update: 2016-11-09

Prepared by:
atsec information security corporation
9130 Jollyville Road, Suite 260
Austin, TX 78759
www.atsec.com

Table of Contents

1 Cryptographic Module Specification 4
1.1 Module Overview 4
1.2 FIPS 140-2 validation 6
1.3 Modes of Operations 7
2 Cryptographic Module Ports and Interfaces 8
3 Roles, Services and Authentication 9
3.1 Roles 9
3.2 Services 9
3.3 Authentication 11
4 Physical Security 12
5 Operational Environment 13
5.1 Applicability 13
5.2 Policy 13
6 Cryptographic Key Management 14
6.1 Random Number Generation 14
6.2 Key / Critical Security Parameter (CSP) Access 15
6.3 Key / CSP Storage 15
6.4 Key / CSP Zeroization 15
7 EMI/EMC 16
8 Self-Tests 17
8.1 Power-Up Self-Tests 17
8.1.1 Integrity Tests 17
8.2 Conditional Tests 18
9 Guidance 19
9.1 Cryptographic Officer Guidance 19
9.1.1 Secure Installation and Startup 19
9.2 User Guidance 19
9.2.1 XTS Usage 19
9.2.2 GCM Usage 20
9.3 Handling Self Test Errors 20
Appendix A Glossary and Abbreviations 21
Appendix B References 22

Introduction

This document is the non-proprietary Security Policy for the Red Hat Enterprise Linux Kernel Crypto API Cryptographic Module v4.0 with CPACF. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-2 (Federal Information Processing Standards Publication 140-2) for a Security Level 1 module.

1 Cryptographic Module Specification

1.1 Module Overview

The Red Hat Enterprise Linux Kernel Crypto API Cryptographic Module v4.0 with CPACF (hereafter referred to as the “Module”) is a software-hybrid cryptographic module that provides general-purpose cryptographic services to the remainder of the Linux kernel. The Red Hat Enterprise Linux Kernel Crypto API Cryptographic Module v4.0 with CPACF is software-hybrid, security level 1 cryptographic module, running on a multi-chip standalone platform.

The module is implemented as a set of shared libraries / binary files.

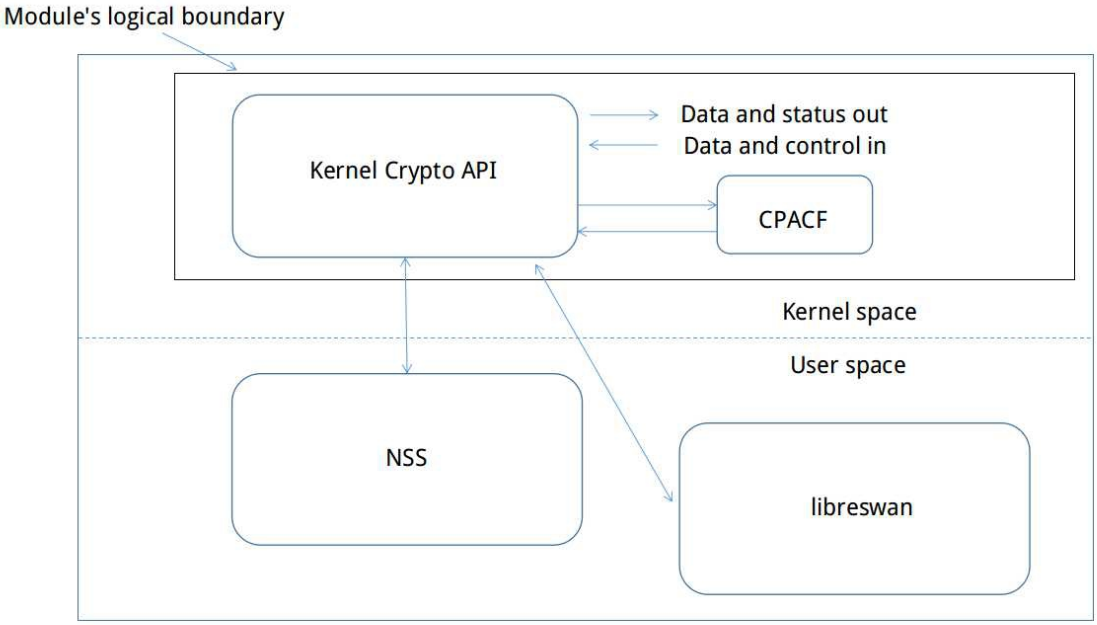

Figure 1: Cryptographic Module Logical Boundary

The module is aimed to run on a general purpose computer; the physical boundary is the surface of the case of the target platform, as shown in the diagram below:

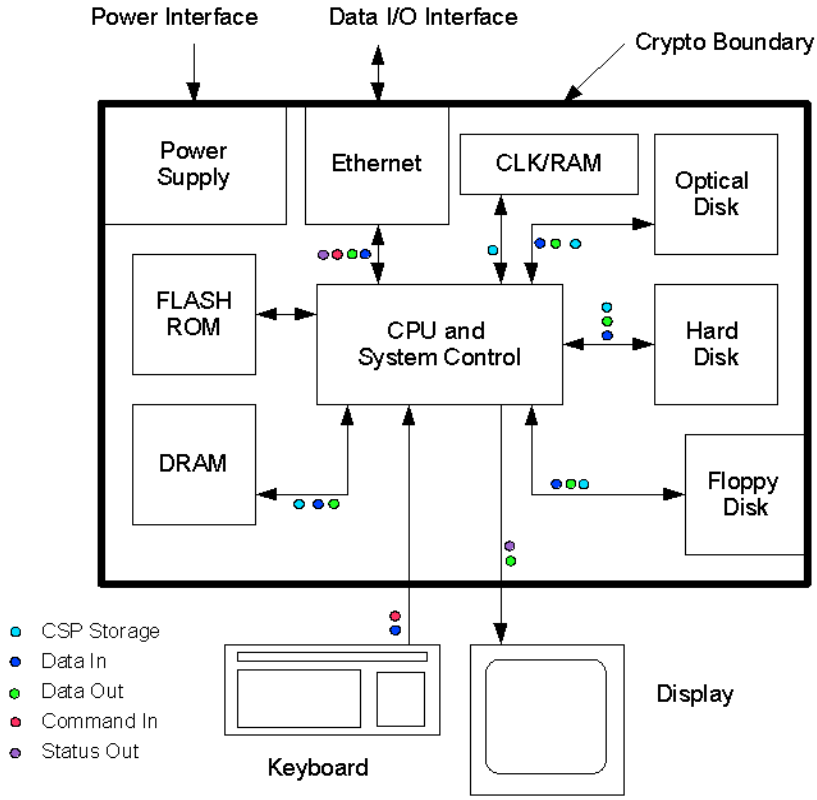

Figure 2: Cryptographic Module Physical Boundary

The list of components required for the module to operate are defined below:

- Red Hat Enterprise Linux Kernel Crypto API Cryptographic Module v4.0 with CPACF with the version of the RPM file 3.10.0-229.11.1.el7
- The configuration of the FIPS mode is provided by the dracut-fips package with the version of the RPM file of 033-241.el7_1.5
- The bound module Red Hat Enterprise Linux NSS Cryptographic Module v4.0 with FIPS 140-2 Certificate #2711 (hereafter referred to as the “NSS bound module” or “NSS module”)
- The bound module Red Hat Enterprise Linux Libreswan Cryptographic Module v4.0 with FIPS 140-2 Certificate #2721 (hereafter referred to as the “Libreswan bound module” or “Libreswan module”)
- The contents of the hmaccalc RPM package (version 0.9.13-4.el7)

The Kernel Crypto API RPM package of the Module includes the binary files, integrity check HMAC files and Man Pages.

The files comprising the module are the following:

- kernel loadable components in /lib/modules/$(uname -r)/kernel/crypto/*.ko
- kernel loadable components in /lib/modules/$(uname -r)/kernel/arch/s390/crypto/*.ko
- static kernel binary /boot/vmlinuz-$(uname -r)
- sha512hmac binary file for performing the integrity checks
- CPACF:
  - Firmware – CP Assist for Cryptographic Functions Feature 3863 (aka FC3863) with System Driver Level 22H

- Hardware – COP chips integrated within processor unit

The NSS bound module provides the HMAC-SHA-512 algorithm used by the sha512hmac binary file to verify the integrity of both the sha512hmac file and the vmlinuz (static kernel binary).

Using of the AES-GCM mode requires the Libreswan module to be bound to this module to satisfy IG A.5 of the FIPS 140-2 Implementation Guidance. The diagram below depicts the relationship between this module and the Libreswan bound module:

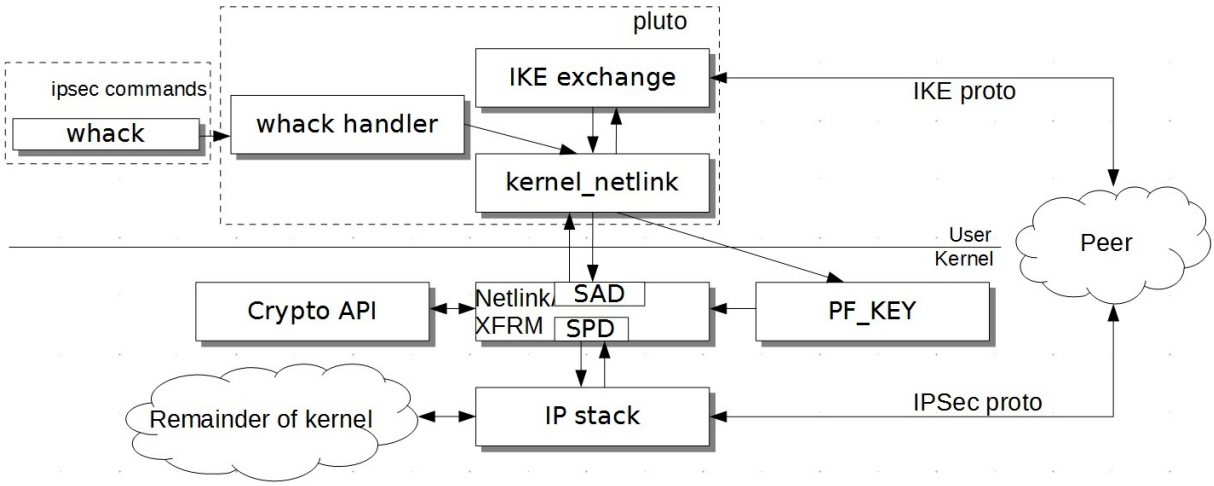

Figure 3: Relationship between Libreswan bound module and Kernel Crypto API module

The Libreswan bound module will provide the IKEv2 protocol implementation that will make the use of the AES GCM implementation of this module by the IPsec protocol. Thus, the operations of the IPsec protocol are entirely within the cryptographic boundary of the module being validated.

The Libreswan module does not need to be bound to the kernel (and thus installed and configured according to its FIPS 140-2 Security Policy) in the case that the AES-GCM algorithm is *not* used.

1.2 FIPS 140-2 validation

For the purpose of the FIPS 140-2 validation, the module is a software-hybrid, multi-chip standalone cryptographic module validated at security level 1. The table below shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2 standard:

FIPS 140-2 Section		Security Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services and Authentication	1
4	Finite State Model	1
5	Physical Security	1
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

Table 1: Security Levels

The module has been tested on the following platforms with the following configuration:

Hardware Platform	Processor	Operating System
IBM z13 mainframe computer	IBM z13	Red Hat Enterprise Linux 7.1

Table 2: Tested Platforms

The physical boundary is the surface of the case of the target platform. The logical boundary is depicted in Figure 1.

1.3 Modes of Operations

The module supports two modes of operation: the FIPS approved and non-approved modes.

Section 9.1.1 describes the Secure Installation and Startup to correctly install and configure the module. The module turns to FIPS approved mode after correct initialization, successful completion of power-on self-tests.

Invoking a non-Approved algorithm or a non-Approved key size with an Approved algorithm as listed in Table 5 will result in the module implicitly entering the non-FIPS mode of operation.

The approved services available in FIPS mode can be found in section 3.2, Table 4.

The non-approved services not available in FIPS mode can be found in section 3.2, Table 5.

2 Cryptographic Module Ports and Interfaces

For the purpose of the FIPS 140-2 validation, the physical ports are interpreted to be the physical ports of the hardware platform on which it runs.

The logical interfaces are the application program interface (API) through which applications request services. The following table summarizes the four logical interfaces:

Logical interfaces	Description	Physical ports mapping the logical interfaces
Command In	API function calls, kernel command line	Keyboard
Status Out	API return codes, kernel logs	Display
Data In	API input parameters	Keyboard
Data Out	API output parameters	Display
Power Input	PC Power Port	Physical Power Connector

Table 3: Ports and Logical Interfaces

3 Roles, Services and Authentication

3.1 Roles

The module supports the following roles:

- **User role**: performs symmetric encryption/decryption, keyed hash, message digest, random number generation, show status
- **Crypto Officer role**: performs the module installation and configuration, module's initialization, self-tests, zeroization and signature verification

The User and Crypto Officer roles are implicitly assumed by the entity accessing the module services.

3.2 Services

The module supports services available to users in the available roles. All services are described in detail in the user documentation.

The following table shows the available services, the roles allowed, the Critical Security Parameters involved and how they are accessed in the FIPS mode. 'R' stands for Read permission, 'W' stands for write permission and 'EX' stands for executable permission of the module:

Service	Algorithms	Note(s) / Mode(s)	CAVS Cert(s).	Role	CSPs	Access
Symmetric encryption/ decryption	Triple-DES	ECB, CBC, CTR	Certs. #1990, #2129 and #2130	User	168 bits Triple-DES keys	R, W, EX
	AES	ECB, CBC, CTR, CCM, GCM, XTS	Certs. #3862, #3863 and #3591		128, 192 and 256 bits AES keys Note: XTS mode only with 128 and 256 bits keys	
		ECB, CCM, GCM	Cert. #3861			
		ECB, CBC, CTR, GCM	Cert. #3570			
Keyed hash (HMAC)	HMAC SHA-1, HMAC SHA-224, HMAC SHA-256, HMAC SHA-384, HMAC SHA-512	BS < KS, KS = BS, KS > BS	Certs. #2508 and #2276	User	At least 112 bits HMAC keys	R, W, EX
Message digest (SHS)	SHA-1, SHA-224, SHA-256, SHA-384 SHA-512	N/A	Certs. #3183 and #2938	User	N/A	R, W, EX
Authenticated encryption	AES CBC mode and HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512	Encrypt-then-MAC cipher (authenc) used for IPsec	Please refer to the AES and HMAC Certs.	User	128, 192 and 256 bits AES keys, HMAC keys	R, W, EX
Random	CTR DRBG	With derivation	Certs.	User	Entropy	R, W, EX

Service	Algorithms	Note(s) / Mode(s)	CAVS Cert(s).	Role	CSPs	Access
number generation (SP 800-90A DRBG)		function, with and without prediction resistance function using AES-128, AES-192 and AES-256	#925, #1095, #1096 and #1097		input string, V, C values and Key	
	Hash DRBG	With derivation function, with and without prediction resistance function using SHA-1, SHA-256, SHA-384 and SHA-512	Cert. #916			
	HMAC DRBG	With and without prediction resistance function using SHA-1, SHA-256, SHA-384 and SHA-512				
Signature verification	RSA	2048 and 3072 bits signature verification according to PKCS#1 v1.5, using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	Certs. #1838 and #1971	Crypto Officer	N/A	R, W, EX
Module initialization	N/A	N/A	N/A	Crypto officer	N/A	EX
Self-tests	HMAC-SHA-512, RSA Signature Verification	Integrity test of the kernel static binary performed by the sha512hmac binary RSA signature verification performs the signature verification of the kernel loadable components	N/A	Crypto officer	HMAC-SHA-512 key	R, EX
Show status	N/A	Via verbose mode, exit codes and kernel logs (dmesg)	N/A	User	N/A	R, EX
Zeroization	N/A	N/A	N/A	Crypto officer	N/A	R, EX
Installation and configuration	N/A	N/A	N/A	Crypto officer	N/A	R, EX

Table 4: Available Cryptographic Module's Services in FIPS mode

In non-Approved mode the Module supports the following non-FIPS Approved algorithms, which shall not be used in the FIPS Approved mode of operation:

Service	Algorithms	Note(s) / Mode(s)	Role	CSPs	Access
Symmetric encryption/ decryption	AES	XTS with 192-bit keys	User	192 bits AES keys	R, W, EX
	DES	ECB (C implementation and CPACF implementation)		56 bits DES keys	

Service	Algorithms	Note(s) / Mode(s)	Role	CSPs	Access
Message digest	SHA-1 (multiple-buffer implementation)	N/A	User	N/A	R, W, EX
	GHASH (C implementation and CPACF implementation)	N/A	User	N/A	R, W, EX
Keyed hash	HMAC	Keys smaller than 112 bits	User	HMAC keys with size less than 112 bits	R, W, EX
Random number generation	ansi_cprng	N/A	User	seed	R, W, EX

Table 5: Service Details for the non-FIPS mode

3.3 Authentication

The module is a Level 1 cryptographic module and does not implement authentication. The role is implicitly assumed based on the service requested.

4 Physical Security

This module is software-hybrid and thus physical security is applicable.

The Red Hat Enterprise Linux Kernel Crypto API Cryptographic Module v4.0 with CPACF inherits the physical characteristics of the host running it. The module has no physical security characteristics of its own. Figure 4: IBM z13 mainframe computer illustrates an IBM System z13 mainframe computer.

Figure 4: IBM z13 mainframe computer

The CP Assist for Cryptographic Function (CPACF) is a hardware device part of the Co-Processor Unit (CoP). It provides full implementations of the DES, Triple-DES, AES, SHA2, SHA3, GHASH, and the SP 800-90A DRBG algorithms. However, the only algorithms that can be used in the FIPS-Approved mode of operation of the module are the algorithms that have been CAVS tested and are claimed as Approved in table 4.

CPACF is made of production grade components and included within the physical boundary of the module, being the IBM z13 mainframe computer.

5 Operational Environment

5.1 Applicability

The module operates in a modifiable operational environment per FIPS 140-2 level 1 specifications. The module runs on a commercially available general-purpose operating system executing on the hardware specified in section 1.2.

5.2 Policy

The operating system is restricted to a single operator (concurrent operators are explicitly excluded). The application that request cryptographic services is the single user of the module, even when the application is serving multiple clients.

In FIPS approved mode, the ptrace(2) system call, the debugger (gdb(1)) and strace(1) shall not be used. In addition, other tracing mechanisms offered by the Linux environment, such as ftrace or kprobes (including systemtap) shall not be used.

6 Cryptographic Key Management

The application that uses the module is responsible for appropriate destruction and zeroization of the key material. The library provides functions for key allocation and destruction, which overwrites the memory that is occupied by the key information with “zeros” before it is deallocated.

6.1 Random Number Generation

The module employs the Deterministic Random Bit Generator (DRBG) based on [SP800-90A] for the creation of random numbers.

The DRBG is initialized during module initialization. The module loads by default the DRBG using HMAC DRBG with SHA-512, with derivation function, without prediction resistance. The DRBG is seeded during initialization with a seed obtained from /dev/urandom of length 3/2 times the DRBG strength. Please note that /dev/urandom is an NDRNG located within the module's physical boundary but outside its logical boundary.

The module performs continuous tests on the output of the DRBG to ensure that consecutive random numbers do not repeat. The module also implements the health checks defined by SP 800-90A, section 11.3. The noise source of /dev/urandom also implements continuous tests.

Here are listed the CSPs/keys details concerning storage, input, output, generation and zeroization:

Type	Keys/CSPs	Key Generation	Key Storage	Key Entry/Output	Key Zeroization
Symmetric keys	AES	N/A	Protected kernel memory	API allows caller on the same GPC to supply key	Memory is automatically overwritten by zeroes when freeing the cipher handler
	Triple-DES	N/A	Protected kernel memory	API allows caller on the same GPC to supply key	Memory is automatically overwritten by zeroes when freeing the cipher handler
DRBG SP800-90A entropy string	SP 800-90A DRBG Entropy string	The seed data obtained from /dev/urandom	Module’s application memory	N/A	Memory is automatically overwritten by zeroes when freeing the cipher handler
SP 800-90A DRBG nonce	SP 800-90A DRBG Seed and internal state values V and C	Based on entropy string as defined in SP 800-90A	Module’s application memory	N/A	Memory is automatically overwritten by zeroes when freeing the cipher handler
HMAC keys	HMAC keys	N/A	Protected kernel memory	HMAC key can be supplied by calling application	Memory is automatically overwritten by zeroes when freeing the cipher handler
Integrity keys	HMAC key used to verify integrity of the sha512hmac file and the static kernel binary	N/A – Installed as part of the module	Persistently stored in plaintext as part of the sha512hmac application	N/A	Zeroized in memory by sha512hmac

	RSA public key used to verify the signatures of kernel objects	N/A – Installed as part of the module	Persistently stored in plaintext as part of the module	N/A	N/A

Table 6: Keys/CSPs

As defined in SP800-90A, the DRBG obtains the entropy string and nonce from the Linux kernel non-deterministic random number generator during:

a. initialization of a DRBG instance

b. after 2^{48} requests for random numbers

The module does not provide any key generation service or perform key generation for any of its Approved algorithms. Keys are passed in from calling application via API parameters.

CAVEAT:

The module generates random strings whose strengths are modified by available entropy.

6.2 Key / Critical Security Parameter (CSP) Access

An authorized application as user (the User role) has access to all key data generated during the operation of the module. Moreover, the module does not support the output of intermediate key generation values during the key generation process.

6.3 Key / CSP Storage

Symmetric keys are provided to the module by the calling process, and are destroyed when released by the appropriate API function calls. The module does not perform persistent storage of keys. The RSA public key used for signature verification of the kernel loadable components is stored outside of the module’s boundary, in a keyring file in /proc/keys/.

6.4 Key / CSP Zeroization

When a calling kernel components calls the appropriate API function that operation overwrites memory with 0s and then frees that memory (please see the API document for full details).

7 EMI/EMC

Product Name and Model: IBM z13

Product Options: All

EMC: Class A

The IBM z13 test platform has “been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.”

8 Self-Tests

FIPS 140-2 requires that the Module perform self-tests to ensure the integrity of the Module and the correctness of the cryptographic functionality at start up. In addition, the module performs conditional test for DRBG.

A failure of any of the self-tests panics the Module. The only recovery is to reboot. For persistent failures, you must reinstall the kernel. See section 9.1 for details.

No operator intervention is required during the running of the self-tests.

8.1 Power-Up Self-Tests

The module performs power-up self-tests at module initialization to ensure that the module is not corrupted and that the cryptographic algorithms work as expected. The self-tests are performed without any user intervention.

While the module is performing the power-up tests, services are not available and input or output is not possible: the module is single-threaded and will not return to the calling application until the self-tests are completed successfully.

8.1.1 Integrity Tests

The Module performs both power-up self tests (at module initialization) and conditional tests (during operation). Input, output, and cryptographic functions cannot be performed while the Module is in a self test or error state. The Module is single-threaded during the self tests and will stop the boot procedure, and therefore any subsequent operation before any other kernel component can request services from the Module.

The Crypto Officer with physical or logical access to the Module can run the POST (Power-On Self-Tests) on demand by power cycling the Module or by rebooting the operating system.

For Known Answer Test, HMAC SHA-512 provided by the NSS bound module is tested before the NSS module makes itself available to the sha512hmac application.

An HMAC SHA-512 (provided by the NSS bound module) calculation is performed on the sha512hmac utility and static Linux kernel binary to verify their integrity. The Linux kernel crypto API kernel components, and any additional code components loaded into the Linux kernel are checked with the RSA signature verification implementation of the Linux kernel when loading them into the kernel to confirm their integrity.

NOTE: The fact that the kernel integrity check passed, which requires the loading of sha512hmac with the self tests implies a successful execution of the integrity and self tests of sha512hmac (the HMAC is stored in /usr/lib/hmaccalc/sha512hmac.hmac).

With respect to the integrity check of kernel loadable components providing the cryptographic functionality, the fact that the self test of these cryptographic components are displayed implies that the integrity checks of each kernel component passed successfully.

The table below summarizes the power-on self tests performed by the module, which includes the Integrity Test of the module itself as stated above and the Known Answer Test for each approved cryptographic algorithm.

Algorithm	Test
AES	KAT, encryption and decryption are tested separately
Triple-DES	KAT, encryption and decryption are tested separately
RSA signature verification	Part of the integrity test (considered as a KAT)
DRBG (CTR, Hash, HMAC)	KAT

HMAC SHA-1, -224, -256, -384, -512	KAT
SHA-1, -224, -256, -384, -512	KAT
Integrity check	HMAC SHA-512

Table 7: Module Self-Tests

8.2 Conditional Tests

The module performs conditional tests on the cryptographic algorithms shown in the following table:

Algorithm	Test
DRBG	The DRBG generates random numbers per block size depending on the underlying DRBG type (CTR, HMAC or Hash based); the 1st block generated per context is saved in the context and another block is generated to be returned to the caller. Each block is compared against the saved block and then stored in the context. If a duplicated block is detected, an error is signaled and the library is put into the “Error" state. The module also performs continuous random number generator test on the output of the NDRNG (/dev/urandom) to check if any two output blocks repeat. If a duplicated block is detected, an error is signaled and the library is put into the “Error” state.

Table 8: Conditional Tests

9 Guidance

This section provides guidance for the Cryptographic Officer and the User to maintain proper use of the module per FIPS 140-2 requirements.

9.1 Cryptographic Officer Guidance

To operate the Kernel Crypto API module, the operating system must be restricted to a single operator mode of operation. (This should not be confused with single user mode which is runlevel 1 on RHEL. This refers to processes having access to the same cryptographic instance which RHEL ensures cannot happen by the memory management hardware.)

9.1.1 Secure Installation and Startup

Crypto Officers use the Installation instructions to install the Module in their environment.

The version of the RPM containing the FIPS validated module is stated in section 1.1 above. The integrity of the RPM is automatically verified during the installation and the Crypto Officer shall not install the RPM file if the RPM tool indicates an integrity error.

To bring the Module into FIPS approved mode, perform the following:

1. Install the dracut-fips package:

```
# yum install dracut-fips
```

2. Recreate the INITRAMFS image:

```
# dracut -f
```

After regenerating the initramfs, the Crypto Officer has to append the following string to the kernel command line by changing the setting in the boot loader:

```
fips=1
```

If `/boot` or `/boot/efi` resides on a separate partition, the kernel parameter `boot=<partition of /boot or /boot/efi>` must be supplied. The partition can be identified with the command "`df /boot`" or "`df /boot/efi`" respectively. For example:

```
$ df /boot
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/sda1	233191	30454	190296	14%	/boot

The partition of `/boot` is located on `/dev/sda1` in this example. Therefore, the following string needs to be appended to the kernel command line:

```
boot=/dev/sda1
```

9.2 User Guidance

CTR and RFC3686 mode must only be used for IPsec. It must not be used otherwise.

When using the Module, the user shall utilize the Linux Kernel Crypto API provided memory allocation mechanisms. In addition, the user shall not use the function copy_to_user() on any portion of the data structures used to communicate with the Linux Kernel Crypto API.

Only the cryptographic mechanisms provided with the Linux Kernel Crypto API are considered for use. The NSS bound module, although used, is only considered to support the integrity verification and is not intended for general-purpose use with respect to this Module.

9.2.1 XTS Usage

The XTS mode must only be used for the disk encryption functionality offered by dm-crypt.

9.2.2 GCM Usage

The GCM mode must only be used in conjunction with the IPSEC stack of the Linux kernel due to the restrictions on the GCM IV generation mandated by IG A.5.

9.3 Handling Self Test Errors

Self test failure within the Kernel Crypto API module or the dm-crypt kernel component will panic the kernel and the operating system will not load.

Recover from this error by trying to reboot the system. If the failure continues, you must reinstall the software package being sure to follow all instructions. If you downloaded the software verify the package hash to confirm a proper download. Contact Red Hat if these steps do not resolve the problem.

The Kernel Crypto API module performs a power-on self test that includes an integrity check and known answer tests for the available cryptographic algorithms.

The kernel dumps self test success and failure messages into the kernel message ring buffer. Post boot, the messages are moved to /var/log/messages.

Use **dmesg** to read the contents of the kernel ring buffer. The format of the ringbuffer (**dmesg**) output is:

```
alg: self-tests for %s (%s) passed
```

Typical messages are similar to "alg: self-tests for xts(aes) (xts(aes-x86_64)) passed" for each algorithm/sub-algorithm type.

Appendix A Glossary and Abbreviations

AES	Advanced Encryption Standard
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining Message Authentication Code
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAT	Known Answer Test
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
NDRNG	Non-Deterministic Random Number Generator
PAA	Processor Algorithm Acceleration
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
TDES	Triple DES
XTS	XEX-based Tweaked-codebook mode with ciphertext Stealing

Appendix B References

FIPS180-4 **Secure Hash Standard (SHS)**
March 2012
http://csrc.nist.gov/publications/fips/fips180-4/fips 180-4.pdf

FIPS186-4 **Digital Signature Standard (DSS)**
July 2013
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

FIPS197 **Advanced Encryption Standard**
November 2001
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

FIPS198-1 **The Keyed Hash Message Authentication Code (HMAC)**
July 2008
http://csrc.nist.gov/publications/fips/fips198 1/FIPS-198 1_final.pdf

SP800-38A **NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques**
December 2001
http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

SP800-38C **NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality**
May 2004
http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated July20_2007.pdf

SP800-38D **NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC**
November 2007
http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

SP800-38E **NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices**
January 2010
http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf

SP800-67 **NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher**
January 2012
http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf

SP800-90A **NIST Special Publication 800-90A - Recommendation for Random Number Generation Using Deterministic Random Bit Generators**
January 2012
http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf

